CLAIMS

1. (**CURRENTLY AMENDED**) A computer-implemented apparatus for handling thread requests in a disparate computer environment, wherein the disparate computer environment arises because the threads requesting services operate asynchronously with respect to each other whereas ~~the~~ service agents servicing the requests operate synchronously with respect to each other, comprising:

a first lock configured to operate as a dispatch lock for allowing ~~that allows~~ only one requesting thread into a dispatch section at a time;

a second lock configured to operate as a service pool lock for synchronizing ~~that synchronizes~~ the requesting thread that is in the dispatch section with a service agent;

wherein, after the requesting thread releases the first and second lock, the service agent handles the request of the requesting thread.

2. (**CURRENTLY AMENDED**) The apparatus of claim **1**, wherein the first lock and second lock are nested relative to each other such that the second lock is acquired while holding the first lock.

3. (**ORIGINAL**) The apparatus of claim **1**, wherein the requesting thread prepares parameters that are to be passed to the synchronized service agent.

4. (**ORIGINAL**) The apparatus of claim **1**, further comprising:

a dispatch module which passes the parameters to the synchronized service agent.

5. (ORIGINAL) The apparatus of claim **4**, wherein the dispatch module operates in the requesting thread's context.

6. (ORIGINAL) The apparatus of claim **4**, wherein the dispatch module selects a service agent that is free from a pool of services.

7. (ORIGINAL) The apparatus of claim **6**, wherein if a free service agent is not available from the pool, then the dispatch module requests and awaits the creation of another service agent that can perform the request of the thread.

8. (ORIGINAL) The apparatus of claim **7** further comprising:

a spawner that creates another service agent based upon the request from the dispatch module.

9. (ORIGINAL) The apparatus of claim **6**, wherein if a free service agent is not available from the pool, then the dispatch module waits on an extant service agent to complete its assignment, wherein the extant service agent is used to service the request of the thread.

10. (**ORIGINAL**) The apparatus of claim **6**, wherein when the service agent completes a request, notification is provided to the waiting requesting thread, and reenters the pool of free service agents in order to await another request from a requesting thread.

11. (**ORIGINAL**) The apparatus of claim **1**, wherein the first lock involves a synchronization point for dispatching the service request.

12. (**ORIGINAL**) The apparatus of claim **11**, wherein the first lock does not involve the requesting thread awaiting completion of a service agent that is handling the request of the thread.

13. (**ORIGINAL**) The apparatus of claim **1**, wherein the service agent is task-based in that services are synchronous.

14. (**ORIGINAL**) The apparatus of claim **13**, wherein the task-based service agent operates in a single-threaded environment.

15. (**ORIGINAL**) The apparatus of claim **13**, wherein the task-based service agent operates in a cooperative multi-tasking environment, wherein only one task-based service agent can execute at a time.

16. (**ORIGINAL**) The apparatus of claim **15**, wherein a first pool of services includes task-based service agents, wherein a second pool of services includes thread-based

service agents, wherein the first and second locks are used in accessing the first and second pools of service agents.

17. (ORIGINAL) The apparatus of claim **1**, wherein utilization of the requesting threads constitutes a technological advance over the use of the service agents.

18. (CURRENTLY AMENDED) The apparatus of claim **17**, wherein the service agents constitute a legacy system which becomes substantially compatible with the requesting threads through use of the first and second locks;

wherein the legacy system includes task-based code that becomes compatible with the requesting thread through utilization of the first and second locks.

19. (**ORIGINAL**) A computer-implemented apparatus for servicing thread requests, comprising:

a first lock that allows only one requesting thread into a dispatch section at a time;

a second lock that synchronizes the requesting thread that is in the dispatch section with a service agent;

wherein, after the requesting thread releases the first and second lock, the service agent handles the request of the requesting thread.

20. (**ORIGINAL**) The apparatus of claim **1**, wherein the requesting thread waits for the results from the service agent handling the request.

21. (**ORIGINAL**) A computer-implemented apparatus for handling thread requests in a disparate computer environment, wherein the disparate computer environment arises because the threads requesting services operate in a multi-threaded processing environment whereas the agents servicing the request operate in a single-threaded processing environment, comprising:

a first lock that allows only one requesting thread into a dispatch section at a time;

a second lock that synchronizes the requesting thread that is in the dispatch section with a service agent;

wherein, after the requesting thread releases the first and second lock, the service agent handles the request of the requesting thread.

22. (**ORIGINAL**) A computer-implemented apparatus for utilizing legacy computer code with multi-threaded code, wherein the multi-threaded code generates service requests, wherein the legacy computer code handles a service request in a single-threaded processing environment, comprising:

a first lock that allows a requesting thread into a dispatch section;

a second lock that synchronizes the requesting thread that is in the dispatch section with a service agent;

wherein the synchronized service agent handles the request of the thread.